Final Report for HPCC/ESS Round II

SGI
Thomas Clune and Spencer Swift

June 29, 1999

Round II of the NASA ESS HPCC Program was a milestone driven, three-way Cooperative Agreement between government, industry and academia. In this three year project, NASA acted as a funding agency to drive technology development in high-performance hardware and scientific software. SGI (formerly Cray Research) was selected as the hardware vendor, which was to provide a suitable high-performance computer and expertise in developing software on the selected platform. Nine internationally recognized Principle Investigators from a variety of disciplines were selected to develop and publish research caliber scientific software capable of running at a sustained 100 gigaflops (GF). These investigators, the codes and their developers are listed in the Table 1. As the vendor component of this triad, SGI's goals were to advance hardware scalability and reliability, to disseminate the technology of high performance programming and to demonstrate future paths in HPCC. Throughout this effort, the close working relationship between SGI and the science researchers has provided us with a unique and valuable vantage point on the project as a whole.

We begin our report with a historical narrative that describes the various tasks that SGI engaged in under the Cooperative Agreement. Following the narrative, we remark on a number of valuable lessons that have been learned by ourselves and ESS as a result of this collaboration. We conclude with a discussion of our perception of the current status of the HPCC effort. Although this project must be considered highly successful by almost any measure and has met some significant requirements of the science community, there remain considerable computational challenges that demand ever increasing computational resources and continuing research into improved algorithms and software design.

Accomplishments

Hardware

One of the primary responsibilities of the vendor for this project was to provide state-of-the art hardware as a development and benchmarking testbed. SGI's strategy was to use a tiered approach to introduce successive generations of technology as it became available. This progression began in late 1996 with the installation of a 512 processor element (PE) T3D. This initial testbed provided 75 gigaflops (GF) peak performance and 32 gigabytes (GB) of main memory to the science teams. The Principal Investigators were expected to demonstrate 10 GF of sustained performance on the provided hardware, and all nine teams were successful in meeting that milestone.

In March of 1997, the Cray T3D was decommissioned and replaced by a 256 PE Cray T3E-600 with a 150 GF peak and 32 GB of main memory. Whereas the processors in the T3D (DEC Alpha EV4's) had no secondary cache, the T3E processors (DEC Alpha EV5's) contained a 96 KB secondary cache, allowing most codes to obtain a significantly larger fraction of peak performance.

SGI's original plan had been to upgrade the T3E with 128 additional pro-

Investigator	Code Name	Researcher(s)	Location
Peter Olson	TERRA	John Baumgardner	Los Alamos
	DYNAMO	Gary Glatzmeier	UC Santa Cruz
Peter Lyster	PSAS	Jay Larson Jing Ghuo	NASA Goddard
	GEOS	Will Sawyer	NASA Goddard
Tamas Gombosi	BATS-R-US	Darren DeZeuw Clinton Groth	The University of Michigan
Graham Carey	MGFLO	Robert Mclay	The University of Texas
Andrea Malagoli	MPS	Anshu Dubey Fausto Cattaneo	The University of Chicago
	HPS	Nic Brummell	The University of Colorado
	PPMMHD	Paul Woodward David Porter	The University of Minnesota
Paul Saylor	CACTUS	Ed Seidel Paul Walker	Max Plank Institute
		Malcolm Tobias Mark Miller Wai-Mo Suen	Washington University
Robert Mechoso	CAMEL	Tony Drummond Joe Sparh John Farrara	UCLA
David Curkendall	SAR	Craig Miller Herb Siegel	JPL
John Gardner	CRUNCH3D	Russ Dahlburgh	NRL
	MHDFCT3D	Rick DeVore	NRL

Table 1: Principle Investigators for Round II of NASA's ESS/HPCC Cooperative Agreement

cessors utilizing a faster clock than the initial 256 PE's (450 MHz vs. 300 MHz). Prior to this planned upgrade however, a hardware issue was discovered in the T3E-600 model computers. Under a well specified, but narrow set of conditions, applications could cause the entire machine to hang. These conditions involved memory access patterns and use of the low level communications hardware. Initially, the bug could be tripped by a code using any of the communication libraries – Message Passing Interface (MPI), Parallel Virtual Machine (PVM) and SHared MEMory access (SHMEM) which is an SGI proprietary paradigm.

The affected hardware was used to pre-fetch (or stream) data from memory upon encountering common memory access patterns. This pre-fetching of data increased the machine's effective memory bandwidth. SGI implemented a fix in the T3E-900 and later model T3E systems, but did not re-engineer the T3E-600. The only way to completely guarantee that these systems would not hang was to disable the affected hardware, i.e. the streaming mechanism. This reduced the effective memory bandwidth and thus lowered the sustained performance of the system below what was specified in the Cooperative Agreement.

To correct this situation, SGI negotiated to upgrade Goddard's system to a 512 PE T3E-600 at no additional cost to NASA. This was in lieu of the 384 PE system with a mixture of T3E-600 and T3E-900 CPUs originally specified in the Cooperative Agreement. Even with streaming hardware disabled, the resulting system met or exceeded the performance parameters specified in the original agreement. An added benefit was that although no code could realistically benefit from faster T3E-900 processors in a heterogeneous system (software typically performs no better than on the slowest processor), the excellent scalability of the T3E allowed all of the investigator codes to benefit from the larger homogeneous system. Thus, Goddard's system became a 512 PE T3E-600 with 300 GF peak and 64 GB of main memory.

In addition, SGI continued development work to mitigate the effect of the hardware bug. The operating system, Unicos/mk, had several design improvements incorporated to make it more fault tolerant of the bug. Instead of hanging the entire machine, only the job running on the affected processor would hang. The Message Passing Interface (MPI) and Parallel Virtual Machine (PVM) libraries were promptly rewritten to recognize these original T3E systems and use a software fix to circumvent the problem. To allow the investigators to continue using the proprietary SHMEM communications library, the site-analysts worked directly with the teams to implement case-specific workarounds to the hardware bug. By default, the streaming mechanism was deactivated for all users, and only activated upon certification by the site analysts - a trivial process, in the case of codes that only used MPI or PVM.

Teams were expected to achieve a sustained 25 GF of performance on this iteration of the testbed and 50 GF on any vendor supplied platform up to twice the performance of the testbed. Subsequent to the renegotiation, this was specified as twice the peak performance of the originally planned 384 PE testbed. With the upgrade, eight teams were able to reach 25 GF in house and four of them were able to achieve 50 GF.

During this time, a large group of Guest Investigators were given small allot-

ments of machine time on the T3E to develop massively parallel software. One of these investigating teams, led by Max Suarez, was particularly successful and was subsequently able to purchase an additional 512 PEs, effectively doubling the T3E's capabilities. Although these additional PEs were primarily slated to serve the NASA Seasonal to Interannual Prediction Project (NSIPP), HPCC was able to utilize all 1,024 processors on multiple occasions. Thus, through a completely unanticipated set of circumstances, the HPCC effort was able to acquire a vastly more powerful testbed than could have been hoped for at the onset of Round II. This final version of the T3E has a 600 GF peak performance and 128 GB of main memory. Upon completion of this project, NCCS will purchase the original 512 PEs (which are still owned by SGI under the terms of the cooperative agreement) and provide the entire installed system to NSIPP, at a substantial cost savings over a standard procurement.

As their final performance milestone, each of the science teams was to provide software that sustained in excess of 100 GF on any existing platform (or set of platforms). With 1,024 processors, Goddard's T3E provided sufficient compute power for the five teams led by Drs. Malagoli, Curkendall, Gombosi, Gardner and Mechoso to reach that major milestone in house.

We feel that it is also important to note that during the course of the project, SGI routinely provided access to extremely large T3E systems in the manufacturing plant prior to customer shipment. The largest of these was a 1,490 node T3E-1200E with a peak performance of almost 1.8 teraflops (TF) and over 350 GB of main memory. Many of the teams were able to benchmark their codes on that system. Several Principal Investigator codes were able to break 200 GF, one broke 300 GF, and one highly optimized spectral code was able achieve in excess of 600 GF on that platform.

In a noteworthy example of inter-agency cooperation, researchers from NERSC, who otherwise only had access to their own 512 PE T3E-900, were able to find and correct a scaling bug in their software using Goddard's 1024 PE system and subsequently run on the large machine mentioned above. This NERSC code was able to exceed 1 TF, and won the 1998 Gordon Bell prize for performance. To our knowledge, this was the first true application to run at over 1 TF sustained performance. In providing this access, NASA was returning a favor to NERSC, which had previously provided ESS with access on their T3E to enable a critical performance milestone to be met on schedule.

Software

Although the evolution of the hardware was perhaps the most visible component of the testbed, the system software also underwent considerable evolution during the project. The stability and robustness of the operating system was increased through several modifications. For example, a PE which "stalls" or crashes can now be "warm" booted rather than completely mapped out of the system. In addition, when a code trips the communication hardware bug, only the processors attached to that job will hang. Previously, the entire machine would hang, including processors and jobs which were not connected to the

offending application. A new tool, PAT (Performance Analysis Tool), was introduced which used direct measurements from hardware counters to measure performance, cache misses, load balance, etc. This was a very nice compliment to the existing tool, Apprentice, which provided source level performance information but was somewhat inaccurate and completely useless for uninstrumented code (e.g. library routines). The strong requirement for absolute compute power also resulted in transfer of optimization obstacles back to the SGI compiler developers in a number of instances.

Site Analysts

In addition to providing the hardware and operating environment for the project, SGI provided two on-site technical analysts who played a critical role in the success of the project. These analysts worked with all nine teams and intimately with eight of them to ensure that performance milestones were met in a timely fashion and in a manner that was acceptable to all parties concerned. Because the selected analysts possessed strong technical backgrounds (in astrophysics and computational fluid dynamics respectively) they were able to communicate effectively with the science researchers while simultaneously tapping into the resources available within SGI. Their varied roles included optimization, algorithm development, implementation advice, bug tracking, and even advocacy in the case of interpreting the intent of the Cooperative Agreement. Even the temporary access to the large off-site machines mentioned above was largely enabled by the presence of on-site staff.

Optimization of the codes from the various Principal Investigator teams was the primary role of the analysts. That effort mostly focused on redesigning the software kernels to attain higher degrees of cache-reuse. A number of codes came directly from vector architectures and were particularly inappropriate for direct optimization via the compiler. Work was also done to exploit some of the T3E-specific features, such as the E-registers, for local and remote strided memory movements (e.g. matrix transposes) and the SHMEM communications library which provides lower latencies than MPI. In some cases, the science teams used the analysts primarily as consultants. For the majority of cases, though, the analysts worked closely with the developers, constructing optimized kernels. Such kernels were always developed in consultation with the original developer or investigating team. After validation, these kernels were then delivered to the investigating team for incorporation in their own source code tree.

In those cases where it became clear that no reasonable amount of code optimization would allow a performance milestone to be met in a reasonable timeframe on the installed testbed, the analysts consulted with personnel at larger T3E sites and SGI manufacturing staff in Chippewa Falls to gain temporary access to machines of sufficient power. Fortunately, the requirement for machines other than the provided testbed was minimal. (10 GF: none, 50 GF: TERRA, MGFLO, MHDFCT3, CRUNCH3D, 100 GF: CACTUS, MGFLO, TERRA)

Guest Investigators

SGI had a vested interest as well as a mandated milestone requirement to assist the community of Guest Investigators mentioned above in successfully porting their codes to the T3E environment. We were pleased to discover that a large majority of those investigators were successful with little or no assistance from the on-site analysts. This speaks directly to the ease-of-use of the T3E as well as to the maturity of the community. The contract requirement to bring ten Guest Investigators to the T3E was easily met.

Future HPCC Architectures

As part of addressing the future of HPCC, SGI was expected to assist the teams in preparing for future generations of hardware. For SGI's roadmap, the successor to the T3E is planned to be the SGI SN1 to be released in the year 2000. Because much of the hardware and operating system of the SN1 is similar to that of the SGI Origin 2000, SGI ported three project codes to that platform. Generally, we have concluded that such porting is straight forward, though the performance relative to that of the T3E can vary dramatically.

Insights

Round II of the ESS/HPCC effort must be considered a major success by a number of measures, and the efforts SGI has expended to ensure that success have identified a number of valuable insights about the strengths and weaknesses of the project as a whole.

Milestone driven performance

Perhaps the greatest strength and the worst liability of the project was the heavy emphasis on raw gigaflop performance. On the positive side, the explicit nature of the metric for success provided clear and challenging goals to the developers and the vendor. Meeting those milestones on the installed testbed then allowed the overarching goals of the project to be greatly surpassed on the more powerful machines available elsewhere. However, the strict nature of these milestones did not adequately reflect differences among the various algorithms nor the initial performance variability of the codes as the teams entered the agreement.

Another strength of the milestone system was that since the individual team milestones were all negotiated with an open time line, the performance milestones were effectively staggered throughout the three year window. This feature helped distribute the workload on the testbed and for the analysts.

Performance Optimization

There is no single feature that was critical to the successful performance of the codes from the participating science teams. Rather, the variety of computational models presented in this round of research displays a variety of techniques used to optimize those models. Some of the optimizations utilized were relevant to most massively parallel (MPP) architectures. Others were specific to the T3E, but usually implemented in a portable way.

Implications of Cache

On both the Cray T3D and T3E, one of the most important hardware aspects which software design should address is the limited primary cache onboard the DEC Alpha chip. Because the latency of moving data from commodity memory to modern RISC processors is hundreds of times longer than the clock interval on those processors, a hierarchy of small high-speed memory, known as cache, is coupled with the CPU to mask the latency. To obtain a substantial fraction of the peak capabilities of a RISC processor, software must reuse a considerable portion of the data in the cache. Fortunately, most scientific software can be designed to at least partially take advantage of cache. On the T3D and T3E, the cache is relatively small compared to other machines of the same generation, making such design particularly challenging.

Cache blocking, the process of designing software for maximal cache reuse is accomplished through breaking the problem up into sections which access and reuse areas of memory no larger than the cache itself. Examples of this include setting appropriate limits on loops, by physically partitioning the problem and by picking a problem size so that it has a natural component the size of the cache.

Because cache is not a feature unique to the T3E, but is common to all RISC based machines and in the future even parallel vector machines, cache blocking will have lasting performance benefits to a piece of software. Since the T3D and T3E have particularly small caches, codes that have been designed for those architectures will likely need little or no cache tuning for other platforms. Nonetheless, where possible, it is best to parameterize kernels in some way to allow portable and well-tuned cache use.

There are two levels of cache on the T3E. The innermost, smallest and fastest is the 8 KB primary cache (or DCACHE), while the outermost, larger and slower secondary cache (or SCACHE) is a 96 KB 3-way set associative cache. Only on rare occasions is it useful to design for both levels of cache. Rather, if an algorithm can exploit the DCACHE, then SCACHE is also being used effectively and can be ignored during the design. Alternatively, if the small size of the DCACHE is prohibitive, considerable performance may still be obtained by designing for the secondary cache. Both situations were encountered frequently in this project.

Memory Access

After cache issues, the next most common performance bottleneck encountered is memory bandwidth. Memory subsystem technology has not kept pace with processor technology. As a result, exploiting modern processor performance requires efficiently accessing memory to achieve near peak memory bandwidth. The Cray MPP computers do not require uniform memory access to achieve maximum floating point performance as on the PVP systems. However, near unit stride references allow single machine load instructions to move more than one data value at a time and allows the read-ahead or "prefetch" hardware to perform as designed. The T3D had a simple, but effective, read-ahead feature. This hardware was redesigned on the T3E to allow prefetching of larger amounts of data and tracking multiple memory reference streams.

Memory access patterns designed to trigger the "streams" reading hardware were beneficial for several of the client codes. A natural consequence of designing memory access patterns for streams activation is that cache use is greatly simplified. In one example, the code TERRA obtained a critical 100% performance gain from such increased bandwidth and corresponding access simplification.

Several hardware issues related to the above discussion came to the forefront during this project. The first was the relatively small primary cache available on the T3E, which increased the difficulty of the task of fitting reasonable size chunks for many of the algorithms. This diminutive cache meant that many problems were limited by memory bandwidth rather than CPU performance. Consequently, codes which were already memory bandwidth limited, could not benefit from later processor upgrades. For example, when these codes were migrated to the T3E-900 and T3E-1200, they showed performance gains of 10% or less over the T3E-600 as compared to the theoretical possibility of a 100% improvement. This limited performance improvement stems from the bottleneck incurred in the memory subsystems which are the same on all these machines. A new memory subsystem with lower latency was introduced in the T3E-1200E. Models which had previously seen a 10% performance gain going from the T3E to the T3E-1200, often obtained a 25% gain between the T3E and T3E-1200E. At the opposite extreme, codes which employed algorithms that efficiently used cache such as matrix-matrix multiplication were able to see more than a 50% performance gain between the high and low ends of the T3E series.

Communication

Because the serial optimizations discussed above benefit performance on any number of processors, only after such issues are addressed is it appropriate to consider communication performance. Here we assume that the code has already been parallelized in some manner, for example, with some sort of domain-decomposition method. Typical serial performance on the T3E-600 was 100 megaflops per processor which is roughly 16% of theoretical peak. All of the codes submitted by the Principal Investigators in this project use some form of explicit message passing. The T3E programming environment does not provide

automated parallelism as on the classic Cray vector machines. For this reason, this NASA project provided the High Performance Fortran compiler from Portland Group, Inc. as an alternative implementation to explicit message passing, but no codes submitted for performance milestones used this. However, some of the Guest Investigators did use that paradigm.

The large majority of the codes used the MPI paradigm with only one implementation using the PVM paradigm. The BATS-R-US model of Tomas Gombosi is a prime example of a code using MPI exclusively and achieving high performance (342 Gflop/s on a 1,490 PE T3E-1200E). While the majority of the Principal Investigators' models were originally developed using the MPI paradigm, some were converted to the SHMEM paradigm or to directly accessing the underlying communications hardware for additional performance. While not portable to other vendors platforms, these code modifications provided significant performance benefits in some cases. The trade off between portability and performance is one that has to be reevaluated on each new machine.

The use of SHMEM is of particular benefit on models with large proportions of communications between all processors as in the case of pseudospectral codes. The low latency aspect of the native communications on the T3E permitted near peak bandwidth on substantially shorter packet lengths than with MPI.

Performance Analysis Tools

Little of this optimization work would have been possible without analysis tools appropriate to the task. The SGI Apprentice tool was useful for both analysts and users to profile a run of an entire model. Apprentice provided high-level aggregate profiling with a source-level interface. A complimentary tool, PAT provided hardware counter profiling for greater precision, but lacked detailed source-level diagnostics. Yet another resource was the hardware counter access library developed by SGI as a part of this project. It requires hand instrumentation of source code but provided key optimization insights on a number of occasions. This is an unsupported product that, none the less, proved essential and has migrated throughout this community.

An interesting example of this was understanding the disappointing performance of the kernel in CACTUS, a general relativity model from Dr. Saylor's team. The algorithm possessed high cache-reuse and yet saw only 15% of peak performance per processor. The hardware counters demonstrated that the problem was the insufficient number (31) of floating point registers on the DEC Alpha EV5. The complex nonlinearities of the various formulae required numerous register "spills", thereby slowing down the code. Although little could be done to ameliorate that bottleneck, cache blocking allowed the kernel to obtain over 90% speedup going to the T3E-1200E.

Another tool that was key to analyzing the Principal Investigator codes was the Godiva tool developed in-house as part of the ESS project. In particular, it was key in identifying the data access pattern and inefficient cache use initially in TERRA. This tool also requires hand-instrumentation but provides far deeper analysis of cache use than available with the tools discussed above.

Machine Utilization

As defined in the cooperative agreement, the investigating teams had some science milestones in addition to the performance milestones discussed thus far. Because of this duality, system usage policy was carefully structured to balance the competing needs. The batch queuing system was set up to allow predominantly development time and short performance benchmarks during the weekdays. Large blocks of time for scientific and benchmarking runs were available on week nights and the weekends.

With user input, the parameters defining the job queue structure were continually adjusted to satisfy the competing needs of the various teams. The success of this effort is a testament to the system administrators careful observations of the dynamic behavior of the queuing load and associated bottlenecks. The administrators were also very adept at manually manipulating jobs to maximize throughput when the automated systems were not up to the task. Further, with the addition of the 512 processors belonging to the NSIPP group, the administrators helped coordinate and configure the machine for dedicated runs on all 1,024 processors. This provided an improved avenue for teams to meet their 50 and 100 GF milestones on site.

At the project onset, Unicos/mk did not support any time sharing model of parallel processes. Later versions of the operating system included Gang Scheduling, a feature which allows up to two parallel jobs to run on the same processors. This flexibility enhancing feature was never enabled on the ESS testbed because of resulting degradation in performance and throughput of applications. While most high performance computing sites are configured to use such a system, the system administration at Goddard appreciated the uniqueness of this site and its requirements and set up the system accordingly.

Conclusion

This project was a successful venture between academia, industry and government. The investigators have all pushed their science and computations to new limits. As the vendor, SGI has significantly increased the robustness and capabilities of the T3E series of computers and assisted users in moving toward future hardware products and software paradigms. Most importantly, NASA has enabled and promoted the use of high performance computational methods for scientific research, especially among its traditional customers in the form of the Guest Investigators.

During the course of this project, the status of massively parallel computer systems as an effective development and production environment for scientific studies has been amply demonstrated. Progressing from the original T3D, to the current T3E-600 and through the off site T3E-1200E, SGI has expanded the capabilities and reliability of these system. The project provided SGI a unique environment to verify and expand our hardware and software. The primary users understood that one project goal was to drive further growth and development

of scalable computing hardware. As intended, this was a cooperative effort and investigators appreciated the fact that along with growth would come some inconveniences.

With the exception of one team's performance milestones, all the team and program milestones have been or will be met by the end of the Round II program in Q4CY99. The reasons for the missed milestones are unrelated to the role of SGI as the hardware vendor and will be addressed in that team's final report.

Even with the wide variety of science and models found in this project, there are a few common conclusions to be made. Primarily, high performance scientific computing can be achieved on scalable systems using explicit message passing models in both an efficient and affordable manner. While this perception is more widely accepted than in years past, this project established new levels of performance and capabilities.

In addition, the variety of science accomplished proves that there is no single ideal method for achieving ever higher performance computing. Clearly explicit message passing works. Either in a portable layered form, such as MPI, or in a proprietary and more direct form, such as SHMEM. Object-oriented methodology is not necessarily an impediment to performance, nor is it always a boon. Models that were written for vector processor or shared memory machines can successfully migrate to distributed massively parallel architectures. Likewise, codes developed using a distributed memory paradigm can migrate to other architectures.

Portability of high performance scientific simulation software is possible with reasonable overhead and performance cost. For a scientist to take advantage of high performance computing, it is not necessary to embrace a single programming model nor throw away an existing one. The key is software engineering and design control. As with most endeavors, careful consideration of the research goals, the design criteria and the available resources streamline this effort.

These achievements also exposed the limitations of current technology both in terms of hardware and in terms of software. Although the majority of codes scaled well beyond the size sufficient to meet the milestones, many did encounter an upper scaling limit on the larger T3E systems. In some cases, the computational abilities have progressed beyond the underlying assumptions inherent in the physical model: the computational models may continue to produce results at higher and higher resolutions, but they often lose any physical significance. Similarly, some fields of study can specify their implementation with ever higher resolution, but have not discovered a numerical scheme which achieves equilibrium within a reasonable time. Implementations may also fail at higher resolutions when condition number of model numbers diverge.

Finally, there are participants who have computational and physical models which scale to larger numbers of processors and higher resolutions than seen during this project. For some of those groups, a significant impediment to progress is the lack of resources. Complete simulations require weeks, months or even years of dedicated time on machines of the caliber of the testbed. To address these and other continually increasing computational needs, continued national support of high performance computing is absolutely necessary. A

follow-on in the form of a Round III would certainly be a positive step in that direction.